说　　明

在毛主席无产阶级革命路线指引下，我国电子工业正在迅速发展。为了适应电子计算机设计和生产的需要，并尽快推广、普及使用电子计算机的技术，本译丛特选编了五篇有关文章，供内部参考。

在程序设计语言方面选了三篇文章，其中两篇综述了至今所论及的语言发展概况，约160余种，附有一览表，并对一些主要的语言如 ALGOL、FORTRAN、COBOL、PL/1等分别作了评价；另一篇是讨论显象用的语言，它是 $ALGOL_{60}$ 的一种扩充，为图象的简单运算，如节点与联线的增加与删除、图象穿程、逻辑加、逻辑乘等而设计的。

在虚拟存贮器方面，选译了两篇文章，介绍了虚拟存贮系统的实现和方法，并对 TSS/360 和 CP—67/CMS 的情况作了描述。

诚然，这几篇文章对计算机专业人员有一定参考价值。但是，作者的评价有其一定的局限性，有的对资本主义国家科学技术还加以宣扬。因此，我们必须本着**“洋为中用”“排泄其糟粕，吸收其精华”**的原则，批判地参看这些文章。

由于编译人员水平有限，工作中错误之处难免，请读者给予批评指正。

编　者

1973.8.

目　　录

原书缺页

得格外重要了。这些语言中大约一半属于“专用语言”的范畴，这些语言在〔18〕中已经作过较为详情的讨论，其它85种语言则有待于作一些约略说明。促使语言领域中这样快速增殖的原因到底是什么？已经发生的事情竟究有什么意义？从1952年开始发展的这数百种语言相互之间的关系——如果有的话，——又是什么？最后，也许是最为重要的是：这些到底预示着什么？由于时间和空间的限制，对于这些问题的回答，不能不有所局限。

在程序设计语言史中，有三点本文不拟加以讨论：一点是关于任何一种具体语言的发展情况，因作者已在〔14〕中对此作过详尽的阐述；第二是具体语言中那套专门的概念；第三，是有关语言的具体实现问题。

1.3 程序设计语言史讨论中的一些问题

在进行程序设计语言史的写作过程中存在几个问题。例如，在对程序设计语言的考虑中，有不少阶段对于整个发展过程都很重要，但在回溯中，要精密地确定其日期又几乎是不可能的。我们——作为一个专业和工业人员——喜欢谈论这些日期，而对于这些日期的函义则不是真正明确的。因此，尽管 COBOL 有着一部最为清晰，记载最为完整的历史，但在确定COBOL的“最早日期”问题上各人仍然提法不一，有人确定为1959年，也有人把它确定为1960年或1961年的。为了确定建立日期，要注意对于程序设计语言至少存在以下几个阶段（事实上，其中的大部分也适用于任何程序）。有些阶段可以是并行发生的。

1．初始的计划；

2．初始计划的第一篇文件；

3．初步的规定；

4．最终的规定（即将要实现的那些规定）；

5．“草样”的运行（即在技巧允许的状况下进行充分调试，但也许并不把所有的特点都包括进去）；

6．“完整语言”的编译程序（或解释程序）进行运行；

7．发展者制订出有关实际问题的习惯用法；

8．除开发展者外，已有若干（关于实际问题）的用户；

9．产生了文件，如手册、指南之类；

10．专业杂志或会议上发表论文（有时这一步与第3步的发生是同样的早）；

11．扩充、修改或新版本。

注意，形式化的定义可能在第3，第4或第9步上出现。还要注意，上述的某些步骤可能同时出现，或组合在一起以致无法分清。不仅如此，第11步还为任一号码较小的步留下了再循环的余地，并对于一个新版本允许重复这个循环。**只有很少的情形才能把上述每一阶段的日期比较容易地取到手。**

写作程序设计语言史中另一方面的问题是有关于一个语言的定义和变形，这会使得精确计数成为不可能的事。纵然可以假定，每当我们看到一种语言时，我们能够识别出它是否是一种较高级语言，但是我们还没有一种严密的方法（或者甚至还没有一种微弱的方法），去判断我们所碰到的语言到底只不过是（标准语言）的一种方言呢？或者确实是一种新的语言。这一点作者在〔17〕中已作了较大篇幅的讨论。在许多具有类似情况

的语言中，作为一个例子，大家可以注意五种模拟语言的存在，SIMSCRIPT Ⅰ，Ⅰ.5，Ⅱ，Ⅱ.5和ⅡPlus。显然，这些语言既相互关联又各不相同。那末它们究竟是五种单独的语言呢？还仅不过是一种语言呢？

2. 语言的历史发展

本节通过一张图表来说明诸语言的历史发展及其相互关系。诸语言的快速增殖及其重要性都指明了。最后，把最重要的一些语言列举出来（这样做法有其相当理由的）。

2.1 按日期的发展情况及相互间的关系

描述程序设计语言的历史发展有三种主要的方式。第一是描述与每种语言有关的诸事件；第二是描述特定年份之内的诸事件；第三是描述诸语言的关系及其后裔。对于第一种方式作者业已做了工作，因此，这里仅希望就第二、第三种方式提供出（有望是）确实的事实基础。这张图表通过提供一个树状网络而总括了第二和第三种方式，图中不仅一望可知地把分年信息描述出来，同时也表明了这些语言的影响及其家世。当然这样一种图表必然会受到空间和图解方面的限制；几乎将任何一部分放大都可能提供出有用的信息。

只有对于相当重要的语言（例如FORTRAN）才标明其独立版本的号码。此图中的图例表明，读者可就某种给定的语言决定此语言出现的最早日期（按作者所知的），另一方面，读者也可依据标明的信息决定某一特定阶段的发展情况（参看1.3）。专用语言（例如模拟，机器工具控制，土木工程）没有列入图中，因为此类语言为数众多，会使复杂性倍增。

此图的背面所列出的语言表现了三个时间点上的快速镜头。仅下列十种语言在所有三张表上都出现（从而表明它们是继续实用的）：ALGOL 60（算法语言60），APT（自动程序控制工具），COBOL（通用商用语言），COMIT（行处理及图形匹配语言），FORTRAN（公式翻译程序），IPL—V（信息处理语言-V），LISP 1.5（表格加工语言1.5），MAD（密执安算法译码程序），MADCAP（数值计算和集合论运算的一种语言），以及NELIAC（海军电子实验室国际算法语言编译程序）。

2.2 语言快速增殖的原因

较高级语言令人难以置信的快速增殖似乎是由下列的一种或数种原因引起的：

1．有一种真正新的语言概念得到了发展，而且/或者是一个新的应用领域认为值得有它自身的语言。

2．在使用一种特定语言取得经验之后，发现其不足之处是那么明显，因而感到为纠正这些缺点而有一种全新的语言，不仅是需要的，而且也是有正当的理由来创造的。

3．几种语言的设施得到了最好的组合而形成一种单独的新语言。

4．感到要想取得新的能力和语言文体上的改变，与其通过对一种现有的语言扩充或修改来达到，还不如用一种新的语言来得容易。

5．设计并实现一种新的语言是一种兴趣，而且也有人愿做这种事，并借以获得基金。

6．即使这些语言中有一种语言可以达到新语言所能达到的目的。但是对于现存的

语言存有个人的偏爱和成见。

7．语言的发展者不知道已经存在一种语言可以满足他的需要，因而他创造了他自己的语言，并相信他满足了上述（1）和（2）中的条件。

对于（1）和（2）略作一点解释，（1）的例子有（但又不局限于）FORTRAN，APT，FLOWMATIC（一种英语式的商业数据处理语言），IPL，GPSS（通用系统模拟程序），COGO（坐标几何）。（2）的主要例子有 COBOL（代替了 FLOW-MATIC）和 SNOBOL（代替了 COMIT）。

2.3 一种语言所以重要的理由

为什么有的语言能得到广泛使用，或者被认为是重要的，或者两者兼之，而另一些语言则对一切实用目的而言，始终只是一小群人的财产？通常，最为明显的答案是：实用性，即对于一类有意义的问题（虽然并不一定是一大类的问题）是适用的，而且又能对此语言写出良好的编译程序。但是，这只是最为明显的表征而已；在这下面还存在着很多别的因素，而这些因素未必全部是以事实为根据的。例如，人们趋炎附势的心理学因素就比许多人想象的重要得多。于是，在一种语言的发展中的有关人物的个人威望和领导权大小就起着巨大的作用。

有些语言开初显然只是星星之火，可是后来这些语言却能无视困难而广为流行——有时竟达到一种狂热的程度。这就相当于“政治上的天赋才智”常常会影响选举结果。发人深省的语言中，最好的例子是 ALGOL 和APL/360（IBM 系统 360 机上的一种程序设计语言）。另一方面，BASIC（初学者的通用符号指令编码）也好（在小的一端），PL/1（程序设计语言/1）也好（在大的一端），都没有引起用户们内心的多大热情，要想精确地找出一种语言中缺乏天赋才智的原因是很困难的，而且，这和实际使用情况很少有关。例如，BASIC 和 COBOL 都是使用极广的语言，但是我很怀疑，许多人对它们是否抱有内心的热情。

总之，一种语言所以被认为重要，实际上有两个主要原因：一是它经济实惠，因而很有用；另一点是它有新的技术。下一节中，我就用了这两个标准。一上来似乎没有任何语言可以满足这两者，这是毫不足怪的。

2.4 重要的语言

在某种意义上，关于下面列出的重要程序设计语言所用的说法几乎对于每个人都是明显的；但在实际上，对于到底应该列出那些语言，任何两个人都未必意见一致！从纯粹的个人角度来说，并适当带有我个人的偏好，我希望把我所认为有重要意义的语言指出来，顺便也说明一下它们之所以重要的理由。

注意：重要性和使用的广泛性并不是一码事；下列语言中有好几种只为相当少的人用过，或者它们仅在一架计算机上实现过。

兹把有重大价值的一些语言按粗略的编年次序列在下面：

APT．这是第一种专用语言。

FORTRAN．这是第一种广泛使用的较高级语言。它为大量科学、工程人员实际使用计算机打开了大门。

FLOWMATIC．这是适于商业数据处理的第一种语言，也是特别强调“类似英语”的句法结构的第一种语言。

IPL—V．表格加工用的第一种，也是主要的一种语言。

COMIT．第一种现实的行处理和图形匹配语言；其大部分特点都出现在（虽然以不同的语法）任何其它试图进行任何行操作的语言中。

COBOL．是一切语言中使用最广泛的语言之一，同时也是商业应用领域使用最广泛的语言，它的技术表征包括对类似英语的语法以及与机器的无关性方面进行了真正的尝试。

ALGOL 60．它以一种精美的形式引进了许多专门的特点，这种语言是以其形式语法定义组合起来的，它对程序设计语言中的大部分理论工作以及对于有关实现技术的许多工作都颇有启发。这种语言在欧洲比在美国应用更为广泛。

LISP．这种语言引进了功能程序设计的概念，并与进行表格加工的设施组合起来。这种语言为在人工智能领域中工作的许多人所使用。

JOVIAL．这种语言是具有足够的能力来处理科学计算，输入/输出，信息的逻辑操作，数据的存贮及处理等诸方面工作的第一种语言。大部分 JOVIAL 编译程序是用 JOVIAL 语言本身写的。

GPSS．这是一种使得模拟技术成为大多数人们的一种实用工具的第一种语言。

JOSS．它是第一种交互性的语言。它派生出相当多的“方言”，这些方言集中起来，使得“分时”对算题而言成为实用。

FORMAC．这是对于需要公式操作的数字问题方面的实际而又相当广泛使用的第一种语言。

APL/360．此语言提供了许多较高级的运算符，从而使得算法缩得非常短，并促成了考察某些问题的若干新途径。

上述所列的重要语言的根本特点是每个语言在某个方面有其独到之处。换言之，每种语言都与独立于所列的其它语言（仅COBOL是例外，因它是大量吸取FLOWMATIC的经验而成的）。进而，每种语言显然是互相独立地发展起来的，虽然每个进行发展的小组（大概）知道在此发展的同时还有其它语言的存在。某些其它语言比上述所列的语言使用更广，而且内容上也更为广泛，尤其是 BASIC，PL/1，SIMSCRIPT 和 SNOBOL。在很多情况中，它们已几乎完全代替了上述所列出某些主要语言（例如，BASIC 代替了 JOSS 及其派生的诸语言，SNOBOL代替了 COMIT等）。刚才提到的四个“明显的候选者”是由于下列原因而未列入重要语言之列：BASIC 虽然简明、经济，但它没有加进新概念，它也不是第一种联机语言，而且就其实际重要性而言，也不是第一等的。PL/1 具有来自 FORTRAN，COBOL 和 ALGOL 的许多能力，用以代替这些语言是它蕴含的目标之一，但这一点没有（或尚未？）成功。并且在把对若干种应用领域的处理能力组合起来的尝试中，JOVIAL 在 PL/1 之先，而SIMSCRIPT 则是建立在以前所有独立的模拟语言的基础上的。还有 SNOBOL 虽然是一种好的语言，但显然只是对 COMIT 中所引进的概念作了相当明显的改善而已。

3．编年发展史

本节的第一部分对主要的几个较早时期进行了描绘，其间的语言要点也指出了。这

就加强和补充了图中载有的信息；本节的第二部分确认在程序设计语言领域中有一些关键概念有别于纯粹的语言发展。本节的第三部分则讨论了作为当前的趋向和活动的语言和课题。

3.1 早年的几个主要时期

3.1.1 最早期（1952年—1956年）。这是一个进行初步探索和试图理解程序设计语言的概念和界限的时期。“伪码”，“自动编码”，“自动程序设计”，“编译程序”，“解释程序”等都是公共的一些术语。在今天的意义之下，伪码单单是指正常机器代码之外的一种语言，自动编码则是用较高级语言进行书写的一般过程，而对于自动程序设计，从直觉上感到这是比自动编译更为高级的某种东西。有趣的是自动程序设计这个术语在冷落了多年之后，又重新流行起来，只是意义略广一些而已。

这一期间所发展起来的所有语言中，仅 FORTRAN（打算用于数值计算的）和 APT（对于机器工具控制的），一直延续到今天，而且两者都经历了许多修正。

3.1.2 里程碑式的会议：1956年，专为讨论较高级语言——或自动编码（这是当时的论题）——的一次十分重要的会议是在1956年于富兰克林学院召开的。对于使用较高级语言的斗争绝没有获胜。没有那个系统在广泛使用。FORTRAN 还没有实现（事实上这次会议没有对 FORTRAN 进行讨论，可是它在1957年西方联合计算机会议上提了出来）。只有一家公司主要在使用 FLOWMATIC 的草样。其它的一些系统主要只是由它们的发展者自己在用。

这次富兰克林学院的会议上，详述和提出了下列语言：

B—O（FLOWMATIC）。这是商业数据处理问题方面的第一种类似英语的语言。仅在尤尼瓦克Ⅰ机上设计并实现。

PRINT Ⅰ。这确实是一种强有力的三地址伪码，其意义在于它能在基本上为商业数据处理设计的机器，即IBM705上提供解数字问题的能力。

OMNICODE。在实质和格式上，它的确是一种类型的汇编语言，然而，它具有对于科学及商用计算的若干强有力的运算。它是为 IBM650 机和 702 机设计的，并极为注意转换潜力。

IT（内部翻译程序）。这是用于数字问题方面的语言，由于打算使用它的 IBM650 机的有限的字符集，因而表示方法上不大灵光。其意义在于它打算在小机器上使用。

Matrix Compiler（矩阵编译程序）。这是一种含有进行矩阵计算的若干操作的较高级语言。它是为尤尼瓦克Ⅰ机设计的。其所以有意义是由于它是第一批专用语言之一（把矩阵操作看作一种专用领域理解）。

NCR 304。这显然是一台机器而不是一种语言。其意义所在是：这显然是首次尝试发展这样的一种计算机，这种计算机由于命令编码处在足够高的级别上，而使得不必进行“自动编码”。（虽然这台机器是明显成功的，但其并没有消除较高级语言的必要性。）

3.1.3 最富有成果的年代：1958年到1959年，在程序设计语言史中，1958 年和1959年似乎显然是最重要的两年。下列事件全部发生在这个期间：

1．IAL（国际代数语言）报告（后成为熟知的 ALGOL 58）的发展和出版。

原书缺页

原书缺页

2．以IAL的规定为基础发展出了三个语言，即NELIAC，MAD和CLIP（Compiler Language for Information Processing），而后者最终成为JOVIAL的基础NELIAC，MAD和JOVIAL至少一直沿用到1971年，而后者主要用于军事方面。

3．在1959年联合国科教组织（UNESCO）会议上提出了描述ALGOL的巴科斯（Backus）形式[5]。从此奠定了程序设计语言中许多理论工作的基础。

4．1959年5月，组成了CODASYL(Conference on Data Systems Languages—数据系统语言会议）短期委员会（Short Range Cormnittee），后来更名为COBOL委员会，并于1959年12月完成详细规定（虽然直到1960年才出版）。

5．AIMACO，Commercial Translator（商用翻译程序）及FACT（Fully Automatic Compiling Technique——全自动编译技术），三种语言的详细规定被发展并可使用。

6．1959年开始了LISP的发展工作。

7．COMIT首次实现（早在1957年12月，其简略说明已出现）。

8．1959年JOVIAL的发展工作开始。

9．IPL—V的一种运行版本1958年初就在IBM650机上使用。1959年夏末一种新版本在IBM704机上运行。

10．对IBM704机发展了APT的第二个版本，即APT Ⅱ。

（还见第11项）。

11．发展了若干种（别的）专用语言，如1958年的DYANA，1959年的DYNAMO；并于1959年开始AED（Automatic Engineering Design—— 自动化工程设计）上的工作。

3.1.4 1960到1970年。这十年是程序设计语言领域的成熟阶段。这一期间，机器编码与其说成是规则，还不如说已成为一种例外，从这个意义上说，使用较高级语言的斗争已显然获胜。甚至用较高级语言来发展系统程序这一点也被完满地接受[3]。诸如ETC（Extendible Compiler——可扩充的编译程序）[7]等强有力的宏系统的使用，以及诸如PC/360等“中介”（half way）语言的使用提供了较高级语言的某些优越性，但对于与机器的无关性问题没有进行任何尝试。

主要的新语言是ALGOL，COBOL和PL/1，其中仅后两种在美国得到重用。虽然ALGOL68已经定义出来，但其实现到70年左右才刚刚开始。

分时的出现，带来了一大群的联机语言，由JOSS带头，BASIC跟随其后，它们都得到很广泛的使用。这两种语言都有许多模仿物和扩充物。APL/360到60年代后期才被利用，并流行于某些专业小组之中。

用于公式操作方面的较高级语言的发展是由FORMAC和Formula ALGOL触发的，虽然仅FORMAC被广泛使用。行处理和图形匹配随着SNOBOL的降临而流行起来。

模拟语言GPSS和SIMSCRIPT使得模拟成为大多数用户可利用的工具，并且也刺激了其它模拟语言的发展。

相当多的专用语言接二连三地发展出来。这方面更详细的情况可参看[14,18]。

FORTRAN和COBOL的正式标准的发展和对PL/1开始标准化也许是最重要的

实用发展之一，尽管此事为许多理论家所看不起。

3.2 与程序设计语言相关联的若干关键概念

为了回顾程序设计语言方面二十年来的工作，有必要把特定语言的历史发展同某些影响所有程序设计语言的概念分别开来，在本文中所述的这些概念与语言中的那些概念是全然不同的。根据粗略的编年次序，我认为主要的概念性发展是（1）形式语言表示法（formal syntactic natation），（2）形式语义定义技术，（3）尝试设计以较高级语言为指令码的机器，（4）用户自行定义的语言。这些概念所以重要的理由由下面来指出。这里我既没有列举较高级语言的概念，也没有列举编译程序的概念，因为它们太基本了，因而认为是当然的。（注：表格加工的概念没有出现在这里，是由于它基本上是一种程序设计技术，因而已在语言中有所表示或包括。这里也没有讨论关于实现中的那些主要概念。）

对于一种程序设计语言的**形式语法表示法**的概念是由巴科斯于1959年引进的[5]。重要的是要注意：BNF（巴科斯范式）只是形式语法表示法的一种表现形式，但不是唯一的一种；例如COBOL的定义就采用了一种不同的但又是等效的元语言，虽然这一事实常常不为人们所了解或为人们所漠视。形式语法表示法有三重意义：

1．它提供了定义语言的语法和消除文句的恼人的含混性（诸如“一个名包含六个字符，并且第一个字符和最后一个字符不能是一个短划号”）的一种严密的方法。

2．它在程序设计语言领域中的实际考虑和语言学家（尤其是却姆斯基（Chamsky）所做的理论工作之间建立了一条纽带。这就可以把语言学的概念和技术应用到程序设计语言方面。虽然BNF是由巴科斯独立发展的，但实际上，仅是却姆斯基语法的一种，只是表示方法不同而已。

3．它还导致面向编译程序的语法的发展，这又依次引起对编译技术更多的理论研究，并产生了发展新的编译程序的“生产线手段”的可能性。

形式语义定义技术远不仅代表超出形式语法定义之外的下一步骤。虽然其许多原始概念是以麦卡锡（McCarthy）的工作为基础的[10]，而对于一种重要语言的形式语义定义的首次实际发展是由IBM的维纳实验室所做的[9]。PL/1的形式定义的庞大规模妨碍了它当前的实际使用。然而，这是**所有**程序设计中一个当前未获解决的问题的一个重要概念性部分，即如何来决定一个程序是否执行了我们所希望它所执行的工作的问题。这一领域中正在取得大量的进展（例如见[1]）。对于程序设计语言还有两个侧面需要考虑：要保证编译程序正确地翻译，要知道为了保证源程序的正确性到底必须做些什么工作。关于解决这两方面的问题，形式语义定义技术都给出了处理方法。

尝试设计以较高级语言为指令码的机器是一种很有意义的概念——虽然至今尚未达到这一目标——因为整个程序设计语言领域明白显示：当前所设计的计算机都非令人满意的口述通讯设备。换言之，我们还没有解决这样的问题，即如何把人们为解决其问题所讲的话，同机器中的物理线路之间的间隙桥接起来。

用户自行定义的语言一直是多年来令人感到兴趣的领域。这方面仅做了一点原始的尝试；这种尝试在[18]中有简略的阐述。允许人们定义他们自己的语言这一点是一个很有意义的概念，因为这就从语言发展者手中，将“对用户有好处”的控制权取出来，并**放到用户手中。**

3.3 当前的现状及课题

在作了回顾之后，为了展望未来，我们首先必须来看看今天。本节的目标是试述1972年程序设计语言方面的现状和主要课题（实现的问题除外）。这些对于未来的潜在影响，将在第四节中讨论。

有关特别**流行的语言**的讨论还在继续，而关于PL/1，ALGOL 68和APL/360的议论最为多见。大量程序设计是用这些语言**之外**的其它语言进行编制这一事实并没有能禁止对它们的讨论。这些议论——至少在公开的议论方面——来自发展者和理论家方面的较来自用户方面的为多。用户继续（当然地）关心成本效用的收益问题和相容性的问题，因而他们喜欢保留使用FORTRAN和COBOL。

最令人惊奇的是语言的快速增殖仍在继续。还没有迹象表明正在创造的语言有所减少。近几年中在美国发展和（或）使用的新语言已在莎美特的[15,16,19]中指出了。

可扩充语言——至少在探索者中间——继续引起兴趣，但是对于业已进行的多年工作，还尚无任何重大价值可见。请参照[4]和[2]中的内容。

证明程序的正确性这一课题正如[1]所证明的那样，正在日益引人注意。这个概念与程序设计语言直接相关，因为所建议的大多数技术都涉及到对程序设计语言改变或补充，或在源程序中插入若干完全不同种类的语句，以使正确性证明能够进行。

把较高级语言用于系统程序设计这一点终于被承认是既正当而又实用的。在辩论和否定了几乎15年之后，真正用较高级语言写的系统程序的数目有了增加；这在[3]中已有了阐述。当今的趋势是集中于不是**该不该**用较高级语言的问题，而是到底用**哪**一种语言的问题。

4. 发展前景

程序设计语言——无论用什么定义——是人们用于同计算机通讯的一种基本手段。从这一前景着眼，一切未来的发展均应致力于使得这种通讯更为简易，更有效用价值。每个进行探索和发展的人都感到他得到了这一答案。而实际的真象似乎是对于人机通讯而言，并不存在一种最好的途径，因而不存在唯一的一种解决办法，**于是也就没有一种适用于任何人的语言。**

本节分为三段——广泛的概念，专用技术和关于当前计算机教育的影响的简要讨论。有关在3.3节中所讨论的若干当前课题的前途的评论也包括在本节之中。

4.1 广泛的概念

应该预期在将来可以看到的一些重要而广泛的概念是：（1）使用自然语言（如英语）；（2）用户自行定义的语言；（3）非过程性的和面向问题的语言；（4）用户计算环境的改善；（5）新的理论性的发展。

人机通讯最终会十分简易，将允许用户用诸如**英语**这样的**自然语言**来指明他的指令——或希望。那是一种速记式的话，用户能用其自然语言包括适用于他的特定领域的记号（如代数式，分子式）。这种概念并非是设想一种能理解全部英语的计算机系统，而是设想许多系统，而每个系统能处理（特别是含有其专用术语的）某个特定的领域。这在概念上就等于人们之间的通讯，只要双方同处在一个领域中而且又能理解同样的话，

这种通讯就能很好进行。承认专用术语的重要性主要是针对歧义性问题的（这种问题本质上存在于专业语汇的**各处**。通过设想一系列的半专用系统，我们就能不再需要一种能理解全部韦氏大字典**外加**一部大百科全书的巨型计算机系统。

赞同和反对这种概念的论点在出版刊物中已多次提到。参考书目能在〔14〕的第九章中找到。

只有到达上述的最终情况时，下一最好的事情才是**用户自行定义的语言**。我们这是指（软件）系统，这种系统首先允许用户定义这样的语言，这些语言能满足他们在功能、专用术语、文体上的个人爱好等诸方面的需要，然后这种系统又要允许能容易地实现这些语言。这个问题的关键部分在于提供这样的一种系统，这种系统易于实现且具有一种可以接受的效能水平。尽管过去的多年来已进行了大量的尝试，但这方面的发展仍然处在一种原始的阶段上。这一方面的参考文献和进一步的讨论可在莎美特的〔18〕和汤姆生（Thompson）与陶斯特（Dostert）的〔21〕中找到。

本文作者经常讲，**非过程性语言**这一术语是随着技术状态的改变而改变的一个相对的术语。因而，当我们发展的语言（及其编译程序）指明的详细信息较少时，我们就提高此非过程性的级。这样一种重要的潜在发展可以单独完成，也可能与**用户自行定义的语言**和（或）**面向问题的语言**结合起来。理想的情况是，用户仅陈述其问题的定义，而由计算机来发展解决办法。虽然要到达能向计算机发出**“计算本公司的工资帐目”**这种要求的日子还至少是未来一、二十年的事，但我相信，我们将会看到，用户必须提供的细节数量会大大减少。更具体地说，我期望关于**做什么**的语句要多一些，而关于**怎样做**的细目要少一些。将会有许多编译程序能有效地决定：对于具体情况，在互不相容的许多算法中到底应选用哪一种。这类研究的例子有ISDOS（见Teichroew和Sayani的〔20〕），NAPSS（Rice的〔11〕）及面向目标的PLANNER（Hewitt的〔8〕）。

拥有今天这样强有力的计算机的用户，其**计算环境**是庞大而复杂的，而且是需要改善的。当用户编写出他的程序时，他在通讯问题上才刚刚开始。这时他必须用命令语言同操作系统接头。借助于由他的错误所产生的内含诊断信息的输出而进行调试，而且要在他不知道还有别人与他同时也在使用这台计算机的情况下试图决定用某种装置的费用。当将来，我们有许多实用的计算机网的时候，甚至他可能不知道他是在用哪台计算机！所有这些困难倾向于否定简化用户问题的直接陈述方面所得到的进展，而这种陈述是用某种较高级语言所写的一个程序来表达的，在当前的环境中，后者只代表了一座冰山浮出水面的顶部，我们当然必须使得这些冰山的其余部分也能比较容易地使用。

事实上，每个人都同意，今天的程序设计是一种技术而不是一门科学。许多人们（虽然为数还较少）争辩说，程序能够（而且应该）成为一门科学。我认为这是可能的，不过还需要许多的**理论性发展**。这些发展范围包括：发展定义程序设计语言的更好的方法，发展一些技术，能使我们知道是否在执行我们真正想要它做的工作，在理解英语方面取得进展，这些对于达到最初所述的目标是需要的。

4.2 未来的具体发展

有很多具体项目，应该对它们在未来将发挥的作用加以评论。在很广的范围中，这些项目与当前感兴趣的许多重要课题是并行的。

在未来的许多年中，**流行语言的使用**似乎不会有剧烈的变化。FORTRAN和COBOL

看来至少和我们共存五年甚至十几年以上。PL/1标准化的潜力，也许会割断FORTRAN和COBOL的某些追随者，但是由于至今对这些较早的语言投资已是如此之巨，以致对于大多数实际问题只要用一种语言，这种优越性也不可能反过来影响过去的历史。ALGOL 68也许会起ALGOL 60所起过的那种作用——即再次启发实现技术的发展和引起对语言的优美性的热忱，然而它对实用界考虑较少。似乎可以确定无疑的是，ALGOL68不会在美国广泛使用，其它地方也许也不会广泛使用。APL/360有不少狂热的信徒，但是尽管他们有此希望和要求，但这种语言（或系统）似乎不会替代所有别的语言。

在可以预见的将来，**专用语言**将继续快速增殖，其理由在[21]中有充分的阐述。实质上，它们表明能给用户带来巨大的经济利益，这来自（1）一种有效的语言（因为这种语言仅处理与实际应用直接有关的东西），（2）若干在某一专业问题范围里早已熟知，并已采用的算法；这些专用语言就数目和有效性而言，还要继续增长，这是由于下述广泛的概念方面技艺的改良：这种概念即提供若干技术以能使用户很方便地定义并实现其自己的语言。作为一个子类，系统程序设计这个应用领域将会看到若干经过改善了的语言继续发展，这些语言将能提供出较高级语言的大部分优点，而且对机器只有最小的依赖性。

近几年来，**可扩充语言**的发展是退步了，说得再好也是停滞不前。对于将来，我希望——但我认为不大可能——这个领域又翻过来，并成为一种有用的工具。然而应该承认：可扩充语言与其说它们本身是一种目标，还不如说它们主要是为达到目标的一种手段。由它们协助而欲达到的目标，即是给人们一种能力，以能简易地定义他们自己的语言。

4.3 日益加强的教育产生的效果

在许多中学里，计算机正在成为一种司空见惯的东西——虽然还说不上是一种平凡的东西，就是在小学里也已成为屡见不鲜。对于下一代的这种教育，要测定其真实的效果（包括直接的和潜在的两个方面），需要有一个比作者所有的更为好的水晶球（指西方算命人用的水晶球——译注）。起码下列几件事情似将发生：（1）人们长大后会把计算机认作为每一个人都要使用的工具，如同汽车、电话以及火炉一样；（2），更多的人将学会使用计算机——诸如BASIC，FORTRAN和COBOL等语言将教给人们，虽然这些语言并非代表程序设计方面，我们已有的最好技术；（3）关于计算机适用范围方面更多的了解，将使得计算机的使用不断增加。

5. 结　语

自1952到1972年的二十年间，曾发展出200种以上的较高级语言。这些语言中，只有13种语言从概念上或从实用观点上看是有重要意义的。1958和1959两年是过去20年中最有意义的年代，1972年的大多数程序设计语言活动都直接或间接地来自那两年所做的工作。当前的**讨论趋向**有PL/1，ALGOL 68和APL/360，而大部分**生产性工作**都是用FORTRAN和COBOL做的。1972年的重要课题是可扩充语言，证明程序的正确性，使用较高级语言于系统程序设计方面，以及继续大量增殖专用语言。将来，可望有的广

泛的概念性发展有：用英语进行程序设计，用户自行定义的语言，更为非过程性的和面向问题的语言，用户计算环境的改善，以及新的理论性的发展。未来具体的发展前景很可能包括继续主要地使用 FORTRAN 和 COBOL，继续大量增殖专用语言。对初、高中学生进行计算机教育的影响是如此深远，以致目前尚无法预测。

参 考 文 献

〔1〕ACM. Proceedings of an ACM Conference on Proving Assertions About Programs. SIGPLAN Notices 7, 1 and SIGACT News 14 (Jan 1972).

〔2〕ACM SIGPLAN. Proceedings of the Extensible Languages Symposium. SIGPLAN Notices 4, 8 (Aug. 1969).

〔3〕ACM SIGPLAN. Proceedings of a SIGPLAN Symposium on Languages for Systems Implementation. SIGPLAN Notices 6, 9 (Oct. 1971).

〔4〕ACM SIGPLAN. Proc. ACM SIGPLAN Conf. on Extensible Languages. SIGPLAN Notices 6, 12 (Dec. 1971).

〔5〕Backus, J. W. The syntax and semantics of the proposed international algebraic language of the Zurich-ACM-GAMM Conference. Proc. International Conf. Information Processing, UNESCO, Paris, 1959, R. Oldenbourg, Munich; Butterworth, London, 1960, pp. 125-32.

〔6〕Bemer, R. W. A politico-social history of ALGOL. In Annual Review in Automatic Programming, Vol. 5, M. Halpern and C. Shaw (Eds.), Pergamon Press, New York, 1969, pp. 151-237.

〔7〕Dickman, B. N. ETC-An extendible macro-based compiler. Proc. AFIPS 1971 SJCC, Vol. 38, AFIPS Press, Montvale, N. J. pp. 529-538.

〔8〕Hewitt, C. Procedural embedding of knowledge in Planner. Proceedings of Second International Joint Conference on Artificial Intelligence, British Computer Society, London, 1971, pp. 167-184.

〔9〕Lucas, P., and Walk, K. On the formal description of PL/1. In Annual Review in Automatic Programming, Vol. 6, 3 Pergamon Press, New York, 1969, pp. 105-182.

〔10〕McCarthy, J. A formal description of a subset of ALGOL. In Formal Language Description Languages, T. B. Steel Jr. (Ed.), North-Holland Pub. Co., Amsterdam, 1965, pp. 1-7.

〔11〕Rice, J. R. On the construction of polyalgorithms for automatic